DL Assignment 2
SRINIVASA REDDY SABBELLA

GitHub link
https://github.com/sabbellasri/Deep-learning/tree/main/hw2

Problem Statement
Create a sequential-to-sequential model to generate captions for a given video, providing a concise description of the actions or events depicted. The model will receive a video as input and produce a continuous stream of text captions that summarize the content of the video.

The development and execution of Recurrent Neural Networks (RNNs) require a robust computational environment, which includes Python as the programming language, CUDA for GPU acceleration, and essential Python libraries such as torch, numpy, scipy, pickle, and pandas to efficiently handle data manipulation, mathematical operations, and model training processes.

Dataset
For the homework assignment, the provided dataset, which includes 1450 training videos and 100 testing videos, was employed to train and test the model. Accompanying the videos, training and testing label.json files were also made available, supplying the necessary labels for the model's training process.

Approach:

For the sequential-to-sequential (seq2seq) model to function effectively, it necessitates the construction of a specific dictionary framework, as outlined in instructional materials. This framework includes special tokens to manage sentence structures: <PAD> for equalizing sentence lengths, <BOS> to signify the commencement of a sentence, <EOS> to indicate the conclusion of a sentence, and <UNK> for representing any words that are not found in the dictionary. Central to this setup are three pivotal python dictionaries. The word_dict compiles the frequency of each word's appearance in the training labels, excluding words that appear fewer than four times to form a refined vocabulary. The w2i dictionary maps each vocabulary word to a unique index, facilitating numerical representation of words for model processing. Conversely, the i2w dictionary reverses this mapping, converting indices back to their corresponding words, ensuring the model's outputs can be translated back into human-readable language. This structure is fundamental for training the seq2seq model, enabling it to learn and generate meaningful sentences based on the input video content.

First of all, we will train a neural network model for a sequence generation task (likely caption generation), using an LSTM encoder and an LSTM decoder with attention, over a specified number of epochs. It preprocesses data, initializes the model and training components, runs the training loop while saving the loss values, and finally saves the trained model to disk. The model has 2 layers namely, encoder and decoder.

This lstm_encoder class defines an LSTM-based encoder module for sequence processing, which first transforms input features from a 4096-dimensional space to a 512-dimensional space using a linear layer, applies dropout for regularization, and then processes the sequence with an LSTM layer. The forward pass reshapes inputs for the linear transformation, applies dropout, and feeds the sequence into the LSTM, returning the LSTM's output along with its final hidden and cell states.

The lstm_attn_decode class defines an LSTM-based decoder with attention for sequence generation tasks, such as caption generation. It includes an embedding layer for input words, dropout for regularization, and an LSTM for sequence processing, integrating attention to focus on different parts of the encoder's output. The forward method generates sequences either by teacher forcing during training (using the true previous output as the next input) or by using its own predictions to feed the next step during inference, dynamically adjusting the teacher forcing ratio based on training steps. The attention mechanism allows the model to weigh different parts of the input sequence differently, aiming to improve the contextuality of the generated sequence. The decoder outputs both the sequence of log probabilities for each word in the vocabulary (used for calculating loss) and the sequence of predicted word indices (used for evaluation or inference).

The attention class defines an attention mechanism in PyTorch, which takes a hidden state and a sequence of encoder outputs as input. It computes attention weights by passing the concatenated representations through a series of linear layers, applies softmax to obtain normalized weights, and then computes a context vector by weighted summing of the encoder outputs, serving as an attentive representation used in sequence-to-sequence models for tasks like machine translation.

And training for certain epochs by adjusting and fine tuning the parameters we can run this model for the test data

Parameters used:

Epochs = 20
Learning Rate = 0.0001
Layers min-Batch Size = 64
Number of Layers that are hidden = 512
Adam Optimiser is used.
dropout regularization. = 0.35

```
python compute_final.py
/home/sabbels/MLDS_hw2_1_data/compute_final.py
```

I got the following results:

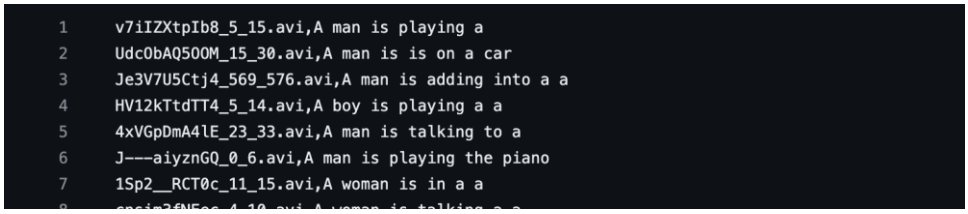
```
Epoch - 20, Batch - 368, Loss - 2.6608142852783203
Epoch - 20, Batch - 369, Loss - 2.8522911071777344
Epoch - 20, Batch - 370, Loss - 2.5698904991149902
Epoch - 20, Batch - 371, Loss - 2.5525190830230713
Epoch - 20, Batch - 372, Loss - 2.378431558609009
Epoch - 20, Batch - 373, Loss - 2.6749846935272217
Epoch - 20, Batch - 374, Loss - 2.747861862182617
Epoch - 20, Batch - 375, Loss - 2.5899622440338135
Epoch - 20, Batch - 376, Loss - 2.468590259552002
Epoch - 20, Batch - 377, Loss - 2.7728209495544434
Epoch - 20, Batch - 378, Loss - 2.6554887294769287
Completed the training
```

The next task is to test the model that I have created a test_func.py to test different videos. I have built the test_func.py using the functions from compute_final.py,
It evaluates the performance of a pre-trained model on a test dataset by generating captions for video content. After loading the model and the word-index mapping (i2w) from files, it processes the test dataset to produce captions, which are then saved to a specified file. Finally, it calculates and prints the average BLEU score of the generated captions against ground truth captions, providing a quantitative measure of the model's captioning performance based on a method described in a referenced academic paper.

I executed the given command on an active node within the Palmetto cluster via SSH. python test.py /home/sabbels/MLDS_hw2_1_data/training_data/feat/ /home/sabbels/MLDS_hw2_1_data/output.txt

the command mentioned above, two arguments are specified: the first argument is the path to the feature map files, which are in the .npy format, and the second argument is the path to the testing_label.json file, containing captions associated with specific video IDs.

Results:



```
1   v7iIZXtpIb8_5_15.avi,A man is playing a
2   UdcObAQ5OOM_15_30.avi,A man is is on a car
3   Je3V7U5Ctj4_569_576.avi,A man is adding into a a
4   HV12kTtdTT4_5_14.avi,A boy is playing a a
5   4xVGpDmA4lE_23_33.avi,A man is talking to a
6   J---aiyznGQ_0_6.avi,A man is playing the piano
7   1Sp2__RCT0c_11_15.avi,A woman is in a a
```

```
[sabbels@skygpug4 MLDS_hw2_1_data]$ ./hw2_seq2seq.sh data/testing_data/feat/ output.txt
Arg 1 -  diff_models(
  (encoder): lstm_encoder(
    (Embedding): Linear(in_features=4096, out_features=512, bias=True)
    (dropout): Dropout(p=0.35, inplace=False)
    (lstm): LSTM(512, 512, batch_first=True)
  )
  (decoder): lstm_attn_decode(
    (embedding): Embedding(2189, 1024)
    (dropout): Dropout(p=0.35, inplace=False)
    (lstm): LSTM(1536, 512, batch_first=True)
    (attention): attention(
      (dense1): Linear(in_features=1024, out_features=512, bias=True)
      (dense2): Linear(in_features=512, out_features=512, bias=True)
      (dense3): Linear(in_features=512, out_features=512, bias=True)
      (dense4): Linear(in_features=512, out_features=512, bias=True)
      (to_weight): Linear(in_features=512, out_features=1, bias=False)
    )
    (to_final_output): Linear(in_features=512, out_features=2189, bias=True)
  )
)
Average bleu score is 0.6911722084486464
```

Upon completing the model training over 20 epochs, I achieved a BLEU score of approximately 0.6911. The predicted captions were subsequently outputted to a file named output.txt.